Conjuntos Parcialmente Ordenados y Retículos

Domingo López Rodríguez

Práctica

Estructuras Algebraicas

Práctica a realizar con R. Se deben seguir las instrucciones y realizar las cuestiones que se piden, entregándolas en formato PDF. Se recomienda utilizar para ello el formato Rmarkdown o Quarto desde RStudio para generar un documento que responda a las cuestiones. Las respuestas a las cuestiones deben estar debidamente razonadas (no vale únicamente con ejecutar código y mostrar la respuesta producida por el programa). Se debe cuidar la presentación y la adecuada expresión escrita.

Contents

Introducción

En esta práctica, usaremos una librería de R que se encuentra alojada en GitHub. Para su instalación, se deben ejecutar los siguientes comandos desde el *prompt* de R:

```
install.packages("remotes")
remotes::install_github("malaga-fca-group/LatticeTheory")
```

> Solo podréis seguir adelante en esta práctica si conseguís instalar esta librería, puesto que el resto de la misma hará uso de los métodos y objetos definidos en ella.

Para poder cargar la librería en memoria, y así poder hacer uso de sus funciones, hay que realizar la siguiente llamada:

```
library(LatticeTheory)
```

En este documento, haremos un breve repaso de la teoría con ejemplos, a la vez que trabajamos dichos ejemplos con R. Las definiciones las encontraremos con fondo azul, las anotaciones con fondo amarillo o rojo, y los ejercicios propuestos con fondo gris.

Posets

Recordemos que un poset (abreviatura de **p**artial **o**rdered **set**, conjunto parcialmente ordenado) es un conjunto X dotado de una relación de orden (parcial) $\leq$ que, por definición, verifica las propiedades siguientes:

- Reflexiva: $x \leq x$ para todo $x \in X$.
- Antisimétrica: si $x, y \in X$ verifican que $x \leq y$ y que $y \leq x$, entonces se tiene que $x = y$.
- Transitiva: si $x, y, z \in X$ verifican $x \leq y$, e $y \leq z$, entonces tenemos que $x \leq z$.

Si, además, se verifica que, para todo $x, y \in X$ se tiene que o bien $x \leq y$ o bien $y \leq x$, entonces se denomina **orden total**.

Una forma *computacional* de expresar una relación de orden es mediante una matriz cuadrada A, donde cada fila y columna representa a un elemento de X y los valores de $a_{i,j}$ son: 1, si el elemento j-ésimo de X es menor o igual que el elemento i; 0 en caso contrario.

Por ejemplo:

```
  a b c d e
a 1 0 0 0 0
b 1 1 0 0 0
c 1 1 1 0 0
d 1 0 0 1 0
e 1 1 1 1 1
```

En este ejemplo, $X = \{a, b, c, d, e\}$ y sabemos que, por ejemplo, $b \leq c$ y $c \leq e$.

Construcción

Para construir un poset usando esta librería, debemos definir primero la matriz mencionada antes.

```
A <- matrix(
  c(
    1, 0, 0, 0, 0,
    1, 1, 0, 0, 0,
    1, 1, 1, 0, 0,
    1, 0, 0, 1, 0,
    1, 1, 1, 1, 1
  ),
  nrow = 5, # Número de filas
  ncol = 5, # Número de columnas
  byrow = TRUE, # Pasamos los valores por filas
  dimnames = list(
    c("a", "b", "c", "d", "e"),
    c("a", "b", "c", "d", "e")
  ) # Nombre de los elementos
)
```

Una vez que tenemos definida la matriz del *orden*, creamos un objeto de tipo `Poset` de la siguiente forma:

```
my_poset <- Poset$new(A)
```

> El nombre de los elementos del conjunto X lo toma de los nombres de las filas y las columnas de la matriz que le pasamos.

Una vez que tenemos este objeto, podemos comprobar su contenido:

```
my_poset
```

```
Poset with 5 elements.
Elements: a, b, c, d, e.
Strict comparabilities:
a < b, a < c, a < d, a < e
b < c, b < e
c < e
d < e
```

Funciones más útiles

A partir de ahora, veremos operadores y métodos definidos sobre los objetos de clase `Poset` que reproducen las operaciones más importantes vistas en teoría.

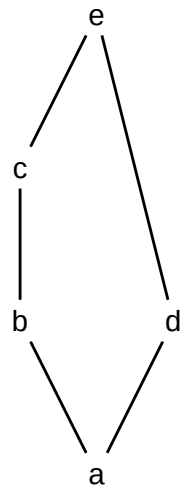
Representación del diagrama de Hasse

La función más visual de todas las que presentaremos, pues es la dedicada a representar gráficamente el diagrama de Hasse de un *poset*.

> El **diagrama de Hasse** de $(X, \leq)$ es el esquema en forma de grafo (X, E), donde cada nodo representa a un elemento de X, y las aristas se corresponden con la relación de orden directa: existe una arista entre a y b si y sólo si $a \leq b$ y no existe ningún elemento *en medio*: no existe c con $a \leq c \leq b$.
> La representación gráfica hace que los nodos se ordenen de abajo hacia arriba siguiendo el orden empleado en el *poset*, de forma que si $a \leq b$, entonces el nodo a estará por debajo de b en la gráfica.

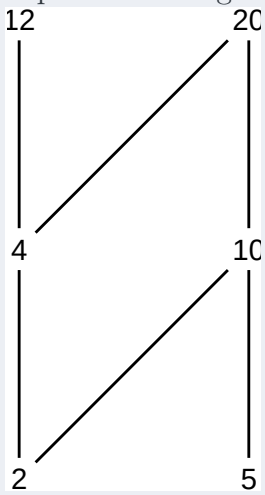
La función a aplicar es `plot()`:

```
my_poset$plot()
```



Ejercicio 1

Escribir el código necesario para reproducir el siguiente diagrama de Hasse:



Elementos destacables

Las funciones que veremos a continuación servirán para determinar los elementos más notables dentro de un subconjunto Y de X.

Elementos maximales y minimales

Dado $Y \subseteq X$, un elemento a es **maximal** si no existe $b \in Y$, $b \neq a$ con $a \leq b$. En otras palabras, si no hay ningún elemento por encima según el orden definido.
De forma análoga, se define elemento **minimal**.

Las funciones para calcular dichos elementos son `maximals()` y `minimals()`. Por ejemplo,

```
my_poset$maximals("a", "b", "d")
```

```
[1] "b" "d"
```

```
my_poset$minimals("b", "c", "d")
```

```
[1] "b" "d"
```

Nota: Fijaos en que los nombres de los elementos son cadenas de caracteres, por tanto deben ir entrecomillados.

Ejercicio 2
Del *poset* del ejercicio anterior, calcular los elementos maximales y minimales de $Y = \{2, 4, 12, 20\}$.

Máximo y mínimo

Dado $Y \subseteq X$, diremos que $y \in Y$ es el elemento **máximo** si $x \leq y$ para todo $x \in Y$, es decir, si y es mayor o igual que todos los elementos de Y.
De forma análoga, se define elemento **mínimo**.

Para calcularlos, tenemos los métodos `maximum()` y `minimum()` :

```
my_poset$maximum("a", "b", "c")
```

```
[1] "c"
```

```
my_poset$maximum("a", "b", "d")
```

```
NULL
```

```
my_poset$minimum("a", "b", "d")
```

```
[1] "a"
```

```
my_poset$minimum("b", "c", "d")
```

```
NULL
```

Como podemos comprobar, para un Y dado no tienen por qué existir los elementos máximos o mínimos. Siempre existirán los maximales y los minimales, pero el máximo o el mínimo, por la definición dada, debe ser único.

Ejercicio 3

Seguimos con el poset de nuestros ejercicios:

a) Calcular el máximo y el mínimo, si existen de $\{2, 4, 5, 20\}$.

b) Calcular el máximo y el mínimo, si existen de $\{2, 4, 12, 20\}$.

c) Dar un ejemplo de conjunto Y que tenga cotas superiores e inferiores pero que no tenga ni máximo ni mínimo.

Cotas superiores e inferiores

Dado $Y \subseteq X$, un elemento $a \in X$ es **cota superior** de Y si $y \leq a$ para todo $y \in Y$.
Es decir, las cotas superiores de Y son elementos de X mayores que todos los de Y.
De forma análoga, se definen las **cotas inferiores**.

Los métodos para estas cotas son `upper_bounds()` y `lower_bounds()`. Por ejemplo:

```
my_poset$upper_bounds("a", "b", "d")
```

```
[1] "e"
```

```
my_poset$lower_bounds("b", "c", "d")
```

```
[1] "a"
```

Ejercicio 4

En el *poset* de nuestros ejercicios:

a) Calcular las cotas superiores e inferiores del conjunto $Y = \{4, 10\}$.

b) ¿Cuáles son las cotas superiores y las inferiores de $\{4, 10, 12\}$?

c) Dar un conjunto que no tenga ni cotas superiores ni cotas inferiores.

Supremo e ínfimo

Al mínimo de las cotas superiores de un conjunto $Y \subseteq X$, si existe, se le denomina **supremo** (es la menor de las cotas superiores).
Al máximo de las cotas inferiores, si existe, se le llama **ínfimo**.

Calculamos estos elementos, si existen, usando las funciones `supremum()` e `infimum()`:

```
my_poset$supremum("b", "c", "d")
```

```
[1] "e"
```

```
my_poset$infimum("b", "c", "d")
```

```
[1] "a"
```

Si el supremo de Y existe y pertenece a Y, entonces coincide con el máximo de Y. Análogamente, si el ínfimo de Y existe y pertenece a Y, entonces coincide con el mínimo de Y.
Y viceversa, si Y tiene máximo, entonces ese elemento es también supremo de Y, y si tiene mínimo, ese elemento es, a su vez, ínfimo de Y.

Ejercicio 5

Dar un ejemplo, en el *poset* que estamos usando en las cuestiones, de:

a) Un conjunto Y_1 que tenga supremo e ínfimo.

b) Un conjunto Y_2 que tenga supremo pero no máximo.

Extensión lineal

Una **extensión lineal** de $(X, \leq)$ es una relación de orden total $\preceq$ definida también sobre X que *preserva el orden* dado por $\leq$, es decir si $a \leq b$ entonces $a \preceq b$.

El método para calcular una extensión lineal (puede haber muchas posibles extensiones) es `linear_extension()`, y devuelve otro *poset*:

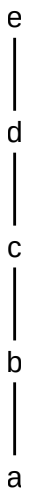
```
extension <- my_poset$linear_extension()
extension
```

```
Lattice with 5 elements.
Elements: a, b, c, d, e.
Strict comparabilities:
```

```
a < b, a < c, a < d, a < e
b < c, b < d, b < e
c < d, c < e
d < e
```

Si nos fijamos, esto es equivalente a $a < b < c < d < e$, y podemos comprobar que es un orden total si dibujamos su diagrama de Hasse:

```
extension$plot()
```



Retículos

Un caso particular de conjunto parcialmente ordenado es lo que denominamos retículo.

> Un **retículo** es un *poset* $(L, \leq)$ donde para cada par de elementos $x, y \in L$ existe su supremo y su ínfimo. Denotaremos:
>
> $$\sup\{x, y\} = x \vee y := \text{supremo de} \{x, y\}$$
>
> $$\inf\{x, y\} = x \wedge y := \text{ínfimo de} \{x, y\}$$

Un retículo se dice **completo** si existe el supremo y el ínfimo de cualquier subconjunto de elementos.

Nota: La notación L proviene de la palabra en inglés para retículo (*lattice*).

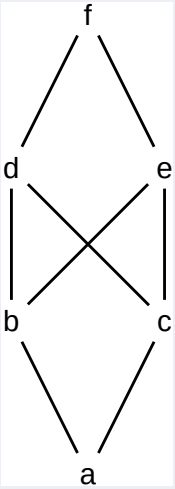
Desde un punto de vista algebraico, podríamos definir un retículo como un conjunto L dotado de dos operadores $\vee$ y $\wedge$ que verifican ciertas condiciones. Sin embargo, en esta práctica estamos interesados en relaciones de orden, así que, de momento, obviaremos esa otra construcción. De todas formas, cuando convenga, utilizaremos la notación $(L, \vee, \wedge)$ para denotar el retículo dotado de esos dos operadores.

Ejercicio 6

¿Tiene estructura de retículo el *poset* que estamos usando en los ejercicios?

Ejercicio 7

¿Tiene estructura de retículo el *poset* dado por el siguiente diagrama de Hasse?



Construcción

Si bien un objeto de tipo `Lattice` se puede construir como un `Poset`, a partir de la matriz que define el orden parcial, existen dos formas sencillas de construirlo bajo condiciones especiales:

a) Si ya tengo un objeto de tipo `Poset`, se le puede aplicar la función `to_lattice()` y, *si es un retículo*, devuelve un objeto de tipo `Lattice`.

b) En la librería, ya hay ciertos tipos de retículos predefinidos que nos pueden ayudar a trabajar, los cuales listaremos en la siguiente sección.

Algunos ejemplos de retículos

Estos tipos predefinidos siempre devuelven objetos de clase `Lattice`.

Cadenas

> Una **cadena** es un conjunto dotado de un orden total.

Por ejemplo:

```
my_chain <- chain(4) # Número de elementos en la cadena
my_chain
```

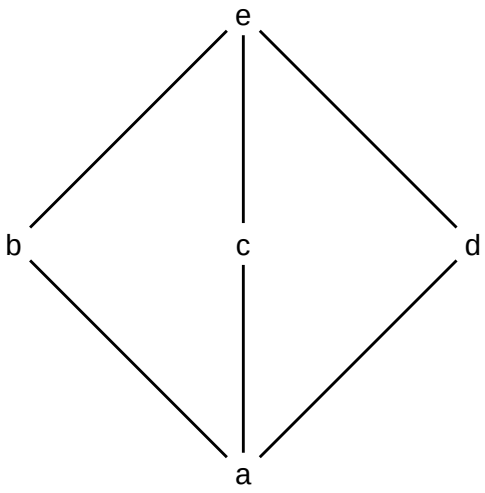
```
Lattice with 4 elements.
Elements: a, b, c, d.
Strict comparabilities:
a < b, a < c, a < d
b < c, b < d
c < d
```

```
my_chain$plot()
```

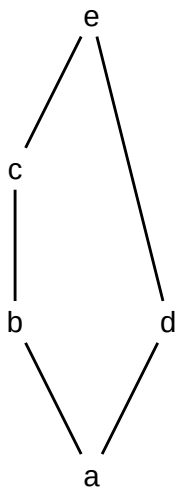
Diamante y pentágono

La forma más fácil de comprender cuáles son estos retículos es ver sus diagramas de Hasse:

```
D <- diamond()
D$plot()
```




```
P <- pentagon()
P$plot()
```



Divisores

> El **retículo de los divisores** de un número entero positivo n se denota por $(\mathcal{D}_n, |)$, donde
>
> $$\mathcal{D}_n := \{m \in \mathbb{Z}^+ : m \text{ divides } n\}$$
>
> y la relación de orden parcial $|$ viene dada por: $m|k$ si, y sólo si, m divide a k.

En este caso, además, podemos ver que los operadores supremo e ínfimo se corresponden, realmente, con las operaciones aritméticas que conocemos desde pequeños: mínimo común múltiplo (mcm) y máximo común divisor (mcd). En otras palabras, $(\mathcal{D}_n, \text{mcm}, \text{mcd})$ es un *retículo algebraico*.

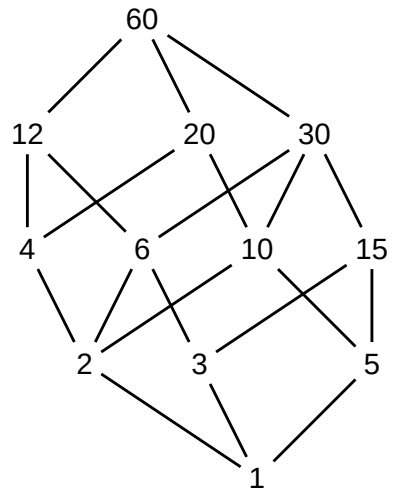
La sintaxis es `divisors_lattice(n)`. Aquí podemos ver un ejemplo:

```
div <- divisors_lattice(60)
div
```

```
Lattice with 12 elements.
Elements: 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60.
Strict comparabilities:
1 < 2, 1 < 3, 1 < 4, 1 < 5, 1 < 6, 1 < 10, 1 < 12, 1 < 15, 1 < 20, 1 < 30, 1 < 60
```

```
2 < 4, 2 < 6, 2 < 10, 2 < 12, 2 < 20, 2 < 30, 2 < 60
3 < 6, 3 < 12, 3 < 15, 3 < 30, 3 < 60
4 < 12, 4 < 20, 4 < 60
5 < 10, 5 < 15, 5 < 20, 5 < 30, 5 < 60
6 < 12, 6 < 30, 6 < 60
10 < 20, 10 < 30, 10 < 60
12 < 60
15 < 30, 15 < 60
20 < 60
30 < 60
```

```
div$plot()
```



Ejercicio 8

Construir el retículo de los divisores de 90 y dibujar su diagrama de Hasse. Éste será el retículo que usaremos en el resto de los ejercicios de esta sección.

Funciones más útiles

Como es un caso particular de *poset*, **todas** las funciones y métodos mencionados en el apartado de Posets siguen siendo válidos. Sin embargo, debido a la naturaleza más re-

stringida de los retículos, se pueden definir otros elementos destacados y otras situaciones más específicas, que veremos a continuación.

Top y bottom

Un retículo $(L, \leq)$ se llama **acotado** si existen los elementos máximo y mínimo de L (equivalentemente, supremo e ínfimo de L). Al supremo de L se le suele denominar **top** y se denota por $\top$ o por 1 (por sus propiedades algebraicas, ya que es el elemento neutro del operador ínfimo). Al ínfimo de L se le denomina **bottom** y se denota por $\bot$ o 0, ya que es el elemento neutro del operador supremo.

Nota: Todo retículo finito es completo y acotado.

Los métodos que nos devuelven estos elementos de un retículo son `top()` y `bottom()`.

Ejercicio 9
¿Cuáles son los elementos $\top$ y $\bot$ del retículo pentágono? ¿Y del diamante? ¿Y del de los divisores de 90?

Átomos y coátomos

Un **átomo** en un retículo $(L, \leq)$ acotado son los vecinos superiores inmediatos del elemento mínimo $\bot$.
Análogamente, un **coátomo** es cada uno de los vecinos inferiores inmediatos del elemento máximo $\top$.

Las funciones correspondientes son `atoms()` y `coatoms()`.

Ejercicio 10

Determina los átomos y coátomos del retículo de los divisores de 90.
¿Qué relación tienen los átomos con los factores primos de 90?

Elementos irreducibles

Un elemento $x \in L$ se llama $\vee$**-irreducible** (supremo irreducible) si no se puede poner como supremo de dos elementos distintos a él.
Un elemento $x \in L$ se llama $\wedge$**-irreducible** (ínfimo irreducible) si no se puede poner como ínfimo de dos elementos distintos a él.

La importancia de estos elementos radica en el **Teorema de representación de Birkhoff**, que nos dice que cualquier elemento del retículo se puede escribir (además, de forma única en un cierto sentido) como el supremo de elementos $\vee$-irreducibles y como el ínfimo de elementos $\wedge$-irreducibles.

La forma sencilla de determinar estos elementos en el retículo, de forma gráfica, es la siguiente:

- Los elementos $\vee$-irreducibles son aquellos que solo tienen un vecino inferior (solo un elemento conectado inmediatamente por debajo).
- Los elementos $\wedge$-irreducibles son los que únicamente tienen un vecino superior.

Las funciones a utilizar para determinar estos elementos son `join_irreducibles()` para los $\vee$-irreducibles y `meet_irreducibles()` para los $\wedge$-irreducibles.

Ejercicio 11

Determina los elementos irreducibles del retículo de los divisores de 90.
¿Qué relación existe entre los elementos $\vee$-irreducibles y la factorización en producto de primos de 90?

Complemento

Un retículo acotado se denomina **complementado** si cada elemento $x \in L$ tiene un complemento, es decir, existe $\hat{x} \in L$ tal que $x \vee \hat{x} = \top$ y $x \wedge \hat{x} = \bot$.

Nota: El complemento de un elemento no tiene por qué existir, y, de existir, no tiene por qué ser único.

El método para determinar los posibles complementos de un elemento dado es `complements()`. Por ejemplo, para calcular los elementos complementarios del elemento d en el pentágono representado más arriba, haremos:

```
P <- pentagon()
P$complements("d")
```

```
[1] "b" "c"
```

Siempre se tiene que el complemento de $\top$ es $\bot$ y viceversa.

Ejercicio 12

¿Existe algún elemento en el retículo de los divisores de 90 que no tenga comple-

mento? ¿Y algún elemento con más de un complemento?

Álgebras de Boole

Una condición algebraica que se puede estudiar sobre los retículos es la distributividad:

Un retículo $(L, \vee, \wedge)$ se dice que es **distributivo** si verifica las leyes distributivas siguientes: para todo $x, y, z \in L$, se tiene

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

Ejemplos de retículos no distributivos son el diamante y el pentágono, antes mencionados. De hecho, un resultado teórico muy importante nos dice cuándo un retículo es distributivo: Un retículo es distributivo si, y sólo si, no contiene subretículos isomorfos al diamante ni al pentágono.

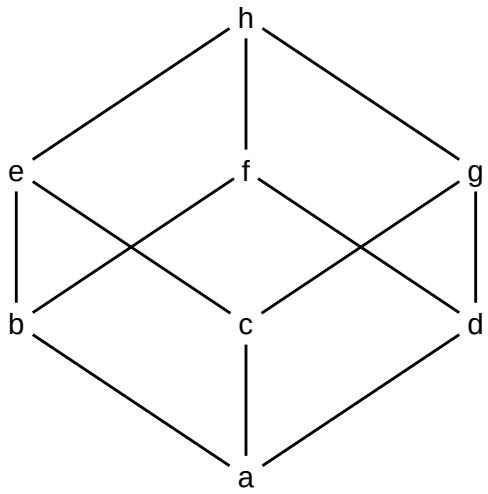
Además, si un retículo es distributivo, podemos asegurar que, si un elemento tiene complemento, éste es único.

Es interesante, por tanto, estudiar retículos complementados y distributivos, y reciben un nombre concreto.

Se denomina **álgebra de Boole** a un retículo acotado, distributivo y complementado.

Para crear un álgebra de Boole con n átomos, se utiliza la sintaxis `boole(n)`.

```
B <- boole(3)
B$plot()
```



Notas adicionales

En esta práctica, nos hemos centrado en operar y razonar sobre retículos y conjuntos parcialmente ordenados, y hemos dejado de lado otras ideas como la de subretículos o la de isomorfismo de retículos (aunque en algún momento las hemos mencionado de pasada). Para completar la parte teórica, es imprescindible considerar el material proporcionado en clase y disponible en el campus virtual.